Introduction to Big Data Assignment-4

Name: Faizan Mulla
Roll No: 21F1003885

Problem Statement

Write PySpark code to implement SCD Type II on a customer master data frame. You will have to create the input files by yourself. For this, use the input data shown as the example in the lecture when this topic was discussed. The PySpark code should be executed on a Dataproc cluster.
Note that within PySpark, there are multiple ways to solve this problem. Any of the ways is acceptable. But do NOT use SparkSQL.

Solution

Environment Setup

1. GCP Setup
   - Set up a new project with default service account configurations
   - Enabled necessary APIs (Compute Engine and Dataproc)

2. Dataproc Cluster Creation
   - Created a Dataproc cluster with following specifications:
      - chose the option: Create cluster on compute engine
      - manager node: series → e2 // machine type → e2-standard-2 (2vCPU, 1 core, 8GB memory)
      - reduce primary disk size from 500GB to something less like 50GB.
      - exact same settings for worker nodes too.

      - Region: **asia-south-1**
      - in the customize cluster menu, **uncheck** the INTERNAL IP ONLY option.

3. Cloud Storage Setup

- Created a Cloud Storage bucket for storing the original and the updated CSV files + python file
- Configured with standard storage class
- Region: **asia-south-1**

Implementation Details

SCD-2.py

```python
from datetime import datetime
from pyspark.sql import SparkSession, functions as F
from pyspark.sql.functions import when


current_date = datetime.now().strftime("%d-%m-%Y")

# Create a spark session
spark = SparkSession.builder.appName("SCD_Type_2").getOrCreate()

# Access data
original = spark.read.csv("gs://iitm-ibd-ga4/original_data.csv", header=True, inferSchema=True)
updated = spark.read.csv("gs://iitm-ibd-ga4/updated_data.csv", header=True, inferSchema=True)

for row in updated.collect():
    condition1 = F.col("name") == row.name
    condition2 = F.col("end_date") > current_date

    original = original.withColumn(
        "end_date",
        when(condition1 & condition2, current_date).otherwise(original.end_date),
    )

for row in updated.collect():
    idx_ = original.tail(1)[0].idx + 1
    name_ = row.name
    dob_ = row.dob
    tuple_ = (idx_, name_, dob_, current_date, "10-12-2099")

    row_ = spark.createDataFrame([tuple_], schema=original.schema)
    original = original.union(row_)

# Show the data
original.show()

# Save to CSV and stop the spark session
original.toPandas().to_csv("output.csv", index=False, header=True)
spark.stop()
```

Execution Process

1. Data Generation / Uploading
   - Uploaded ***original_data.csv*** & ***updated_data.csv*** files to Cloud Storage Bucket.
   - Changed the file names in the Python script to the gsutil URI of the respective files.

2. SCD-2
   - Now, upload ***SCD-2.py*** to Cloud Storage
   - Created Dataproc Cluster
   - Submitted Spark job through Dataproc
   - Monitored job execution through Dataproc UI

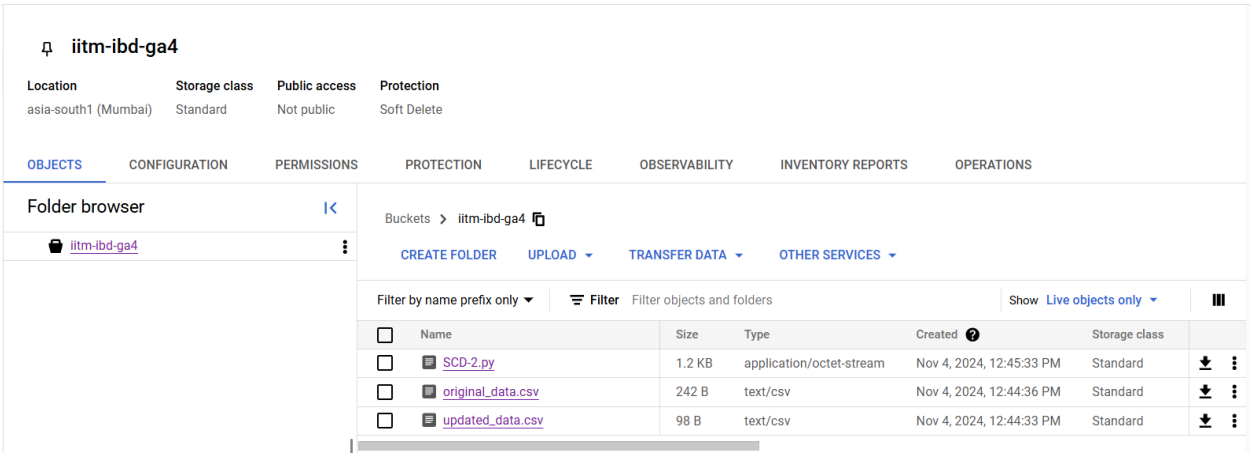
Results

output.txt

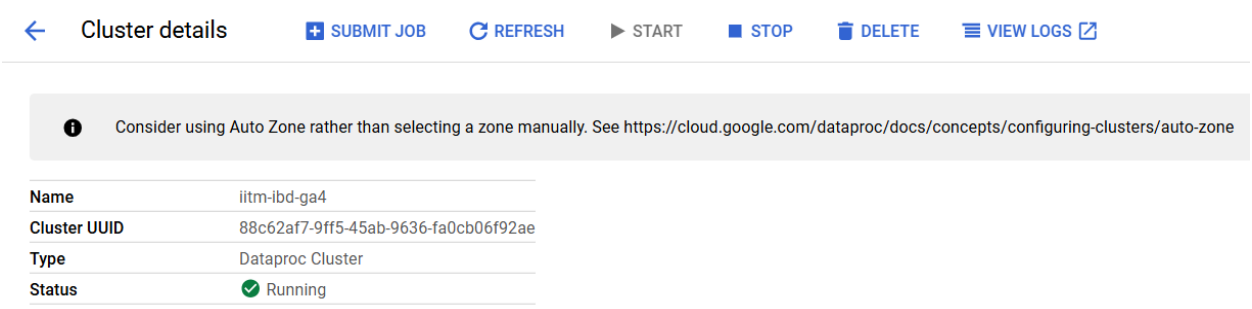
```
+---+-------+----------+----------+----------+
|idx|   name|       dob|start_date|  end_date|
+---+-------+----------+----------+----------+
|  1|  Alice|1990-01-01|01-01-2023|04-11-2024|
|  2|    Bob|1985-05-15|01-02-2023|10-12-2099|
|  3|Charlie|1992-09-09|01-03-2023|10-12-2099|
|  4|    Eve|1987-12-12|01-04-2023|04-11-2024|
|  5|  Frank|1993-06-22|01-05-2023|10-12-2099|
|  6|  Alice|1990-01-01|04-11-2024|10-12-2099|
|  7|  David|1988-08-08|04-11-2024|10-12-2099|
|  8|    Eve|1987-12-12|04-11-2024|10-12-2099|
|  9|  Grace|1991-10-10|04-11-2024|10-12-2099|
| 10|  Henry|1995-03-03|04-11-2024|10-12-2099|
+---+-------+----------+----------+----------+
```

Relevant Screenshots

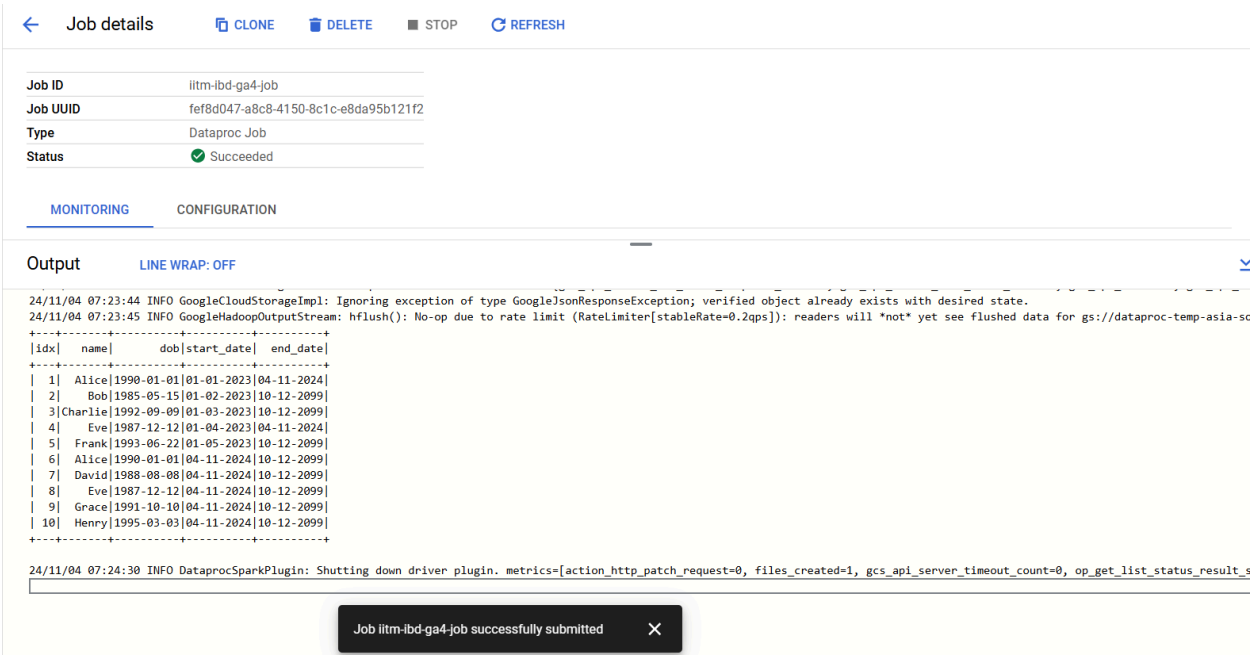
1. Cloud Storage Bucket Contents



2. Dataproc Cluster Configuration



3. **Job Execution Results**



4. **Output File Contents**